(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 876 027 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.11.1998 Bulletin 1998/45

(51) Int Cl.[6]: **H04L 9/08**

(21) Application number: **98303315.0**

(22) Date of filing: **28.04.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **30.04.1997 US 846973**

(71) Applicant: **Sun Microsystems Inc.**
Palo Alto, California 94303-4900 (US)

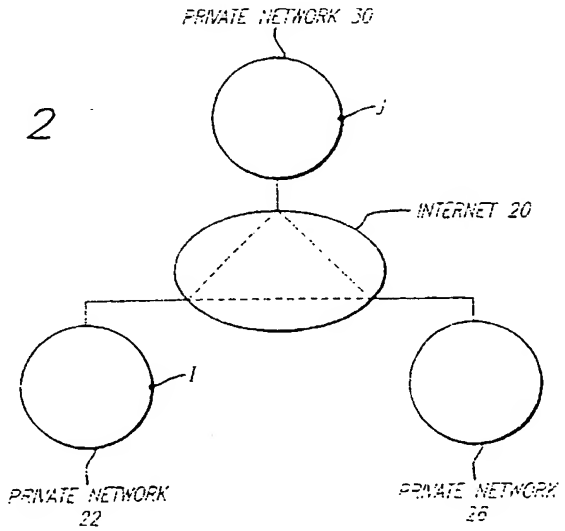
(72) Inventor: **Aziz, Ashar**
10/4, Islamabad (PK)

(74) Representative: **O'Connell, David Christopher**
Haseltine Lake & Co.,
Imperial House,
15-19 Kingsway
London WC2B 6UD (GB)

(54) Method and apparatus for achieving perfect forward secrecy in closed user groups

(57) A method and apparatus for achieving perfect forward security for closed user groups having X nodes is disclosed. A first data processing device node I (22) is coupled to a private network which is in turn coupled to the Internet (20). A second data processing device node J (30) is coupled to the same, or to a different network, which is also coupled to the Internet (20), such that node I (22) communicates with node J (30) using the Internet protocol. A unique secret value and a public value to each of the X nodes is provided. A context variable N_i is also provided to each of the X nodes. Each of the X nodes obtain a certificate for other (X-1) nodes and obtains the other (X-1) nodes' public value (μ^i mod p) from the certificate. Each node in a closed user group precomputes all shared secrets (e.g., $\mu^{(2N)ij}$ mod p) for each closed group node. Each node then deletes its secret (i, j, ..., etc.). As the value of N is incremented, each node may compute $[K_{ij}]_N$ for any value of N without the need to recalculate the shared secret. Upon receipt of a data packet in the context N_i (for a datagram transmitted, for example, by node I), a receiving node J computes $[K_{ij}]_{N_i}$ and decrypts the data packet using $[K_{ij}]_N$. Since it is not necessary to compute the implicit shared secret between the nodes in the closed user group, perfect forward secrecy is achieved since a cracker's discovery of the value of $[K_{ij}]_{N_i}$, and thereby the value for a particular N of $\mu^{(2N)ij}$ mod p, will not assist in decrypting packets encrypted in contexts earlier than N_i.

FIG. 2



EP 0 876 027 A2

Description

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates to the field of key management schemes, and more particularly, the present invention relates to a key management scheme for Internet working protocols to provide additional security at the network layer.

2. Art Background:

The Internet comprises a spiderweb of connected networks which criss-cross the globe and permit users to send and receive data packets between computers. Although many of the computers coupled to the Internet are disposed at fixed locations, portable computer systems may be physically moved from one location on a network to another. Wireless links coupling the computers to the Internet, including direct satellite links, also allow users to access the Internet from remote areas. As a result of the dramatic increase in the use of the Internet throughout the world, concerns regarding network security naturally arise.

A variety of schemes have been proposed to increase security on the Internet, and a number of these schemes have been adopted. For example, encryption and authentication procedures known as Privacy Enhanced Mail (PEM) provide for enhanced privacy in electronic mail ("e-mail") services over the Internet. Additionally, schemes for utilizing PEM for secure remote user authentication have also been proposed.

However, even if a remote user has been authenticated, there still exists the possibility that an intruder (herein referred to as a "cracker") may mount an active attack to interject himself in data transfers across the Internet. Although a user may incorporate a scheme for secure remote user authentication prior to login, a cracker may sever one of the authenticated parties from the Internet connection, and receive and transmit substitute data packets to the other unwitting party (or potentially to both parties). Once the Internet connection is established, data packets are sent over the network in the clear. For example, a cracker may interject himself between, for example, a user "A" in communication with a user "B" on the Internet, and issue a disconnect command to user A. Upon receipt of the disconnect command from the cracker, user A believes that user B has severed the connection. The cracker may then take over the communication established with user B, such that user B does not know that user A is not sending him data packets. Thus, a number of security issues exist when sending data over the Internet, including a cracker's ability to monitor data packets in the clear and to interject himself in the communication line such that he may receive and send data packets to unwitting users. It is, therefore, advantageous to put authenticity and privacy features at the network layer on the Internet. However, the majority of the privacy and authentication protocols which have been proposed provide session oriented key management schemes. Unfortunately, many of the commonly used network layer protocols are session-less datagram oriented protocols.

One example of a simple key management scheme is referred to as "SKIP" for use in session-less datagram protocols. In the SKIP scheme, a first data processing device (node I) is coupled to a private network which is in turn coupled to the Internet. A second data processing device (node J) is coupled to the same, or to a different network, which is also coupled to the Internet, such that node I communicates to node J using the Internet protocol ("IP"). Node I is provided with a secret value i, and a public value μ^i mod p. Node J is provided with a secret value j, and a public value μ^j mod p. Data packets (referred to as "datagrams") are encrypted using the teachings of the present invention to enhance network security. A source node I obtains a Diffie-Helman (DH) certificate for node J (either from a local cache, from a directory service, or directly from node J), and obtains node J's public value μ^j mod p from the DH certificate. Node I then computes the value of μ^{ij} mod p, and derives a key K_{ij} from the value μ^{ij} mod p. A transient key K_p is generated at random and is used to encrypt the datagram to be sent by node I. The key K_p is used for a configurable number of bytes, which is the maximum number of bytes the node will encrypt using K_p. The key K_p is then encrypted with key K_{ij}.

Upon receipt of the encrypted datagram by the receiving node J, the node J obtains a DH certificate for node I (either from a local cache, from a directory service or directly from node J) and obtains the public value μ^i mod p. Node J then computes the value of μ^{ij} mod p and derives the key K_{ij}. Node J utilizes the key K_{ij} to decrypt the transient key K_p, and using the decrypted transient key K_p, node J decrypts the datagram packet, thereby resulting in the original data in unencrypted form.

One aspect of the SKIP scheme is that K_{ij} stays constant until the DH certificate changes. Depending on the environment, obtaining a new DH certificate may result in system performance degradation. As will be described, the present invention discloses a method and apparatus for generating other implicit keys from K_{ij}, without the necessity of generating a new DH certificate or requiring any communication between node I and J to change keys. Using the teachings of the present invention, one secret may be used to generate literally millions of secret keys by stepping the

context, where a context is defined by an implicit interchange key. In addition, the present invention provides methods and apparatus for achieving perfect forward secrecy in closed user groups, through the application of one-way functions to the implicit pair-wise secrets for each node. Moreover, the present invention discloses an improved application of SKIP for datagram multicasts.

SUMMARY OF THE INVENTION

The present invention provides an improved simple key management scheme (SKIP) having particular application to datagram protocols, such as the Internet protocol (IP).

In one embodiment, the present invention discloses methods and apparatus for achieving perfect forward security in closed user groups having X nodes. A first data processing device (node I) is coupled to a private network which is in turn coupled to the Internet. A second data processing device (node J) is coupled to the same, or to a different network, which is also coupled to the Internet, such that node I communicates with node J using the Internet protocol. A unique secret value and a public value to each of the X nodes is provided. A context variable N_i is also provided to each of the X nodes. Each of the X nodes obtain a certificate for other (X-1) nodes and obtains the other (X-1) nodes' public value (μ^j mod p) from the certificate. Each node in a closed user group precomputes all shared secrets (e.g., $\mu^{(2N)ij}$ mod p) for each closed group node. Each node then deletes its secret (i, j, ... etc.). As the value of N is incremented, each node may compute $[K_{ij}]_N$ for any value of N without the need to recalculate the shared secret. Upon receipt of a data packet in the context N_i (for a datagram transmitted, for example, by node I), a receiving node J computes $[K_{ij}]_{N_i}$ and decrypts the data packet using $[K_{ij}]_N$. Since it is not necessary to compute the implicit shared secret between the nodes in the closed user group, perfect forward secrecy is achieved since a cracker's discovery of the value of $[K_{ij}]_{N_i}$, and thereby the value for a particular N of $\mu^{(2N)ij}$ mod p, will not assist in decrypting packets encrypted in contexts earlier than N_i.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a data processing system incorporating the teachings of the present invention.

Figure 2 diagrammatically illustrates one possible network scheme using the teachings of the invention in an Internet environment.

Figure 3 illustrates a flow chart of the steps executed in sending an encrypted data packet from a network node I to a network node J, in accordance with the teachings of the present invention.

Figures 4(a) and **4(b)** are flow charts of the steps executed by each node to change the value of the variable N in the calculation of $[K_{ij}]_N$ and to decrypt a data packet where $N_i < N_j$.

Figures 5(a) and **5(b)** are flow charts of the steps executed for the receipt of encrypted data packets by node J from node I where $N_i = N_j$ and $N_i < N_j$.

Figure 6 diagrammatically illustrates the transmission format of an encrypted datagram.

Figure 7 is a flow chart of the steps executed by the present invention to achieve perfect forward secrecy for closed user groups.

Notation and Nomenclature

The detailed descriptions which follow are presented largely in terms of symbolic representations of operations of data processing devices coupled to a network. These process descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities may take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, displayed and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, operations, messages, terms, numbers, or the like. It should be borne in mind, however, that all of these similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

In the present invention, the operations referred to are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers (referred herein as "nodes"), or other similar devices. In all cases, the reader is advised to keep in mind the distinction between the method operations of operating a computer and the method of computation itself. The present invention relates to method steps for operating a computer, coupled to a series of networks, and processing electrical or other physical signals to generate other desired physical signals.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purposes or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. The method/process steps presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein. or it may prove more convenient to construct specialized apparatus to perform the required method steps. The required structure for a variety of these machines will be apparent from the description given below.

Detailed Description of the Invention

In the following description, numerous specific details are set forth such as system and network configurations, representative data packets, messages, and devices, etc., to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well known circuits and structures are not described in detail in order to not obscure the present invention. Moreover, certain terms such as "knows", "verifies", "examines", "utilizes", "finds", "determines", "challenges", "authenticates", etc., are used in this Specification and are considered to be terms of art. The use of these terms, which to a casual reader may be considered personifications of computer or electronic systems, refers to the functions of the system as having human-like attributes, for simplicity. For example, a reference herein to an electronic system as "determining" something is simply a shorthand method of describing that the electronic system has been programmed or otherwise modified in accordance with the teachings herein. The reader is cautioned not to confuse the functions described with everyday human attributes. These functions are machine functions in every sense.

Exemplary Hardware

Figure 1 illustrates a data processing system in accordance with the teachings of the present invention. Shown is a computer 10, which comprises three major components. The first of these is an input/output (I/O) circuit 12 which is used to communicate information in appropriately structured form to and from other portions of the computer 10. In addition, computer 10 includes a central processing (CPU) 13 coupled to the I/O circuit 12 and a memory 14. These elements are those typically found in most general purpose computers and. in fact. computer 10 is intended to be representative of a broad category of data processing devices. Also shown is an interface circuit 17 coupled to the I/O circuit 12 for coupling the computer 10 to a network, in accordance with the teachings herein. The interface circuit 17 may include encrypting and decrypting circuitry incorporating the present invention, or as will be appreciated, the present invention may be implemented in software executed by computer 10. A raster display monitor 16 is shown coupled to the I/O circuit 12 and issued to display images generated by CPU 13 in accordance with the present invention. Any well known variety of cathode ray tube (CRT) or other type of display may be utilized as display 16.

Referring now to **Figure 2,** a simplified diagram conceptually illustrates the Internet 20 coupled to a private network 22, a second private network 26, and a third private network 30. The network topology illustrated in **Figure 2** is representative of the existing Internet topology, however, it will be noted that the present invention provides an improved key management scheme which has application for use in networks other than the Internet.

One of the unique aspects of the Internet system is that messages and data are transmitted through the use of datagram packets. In a datagram-based network, messages are sent from a source to a destination in a similar manner to a government mail system. For example, a source computer may send a datagram packet to a destination computer regardless of whether or not the destination computer is currently on-line and coupled to the network. The Internet protocol (IP) is completely session-less, such that IP datagram packets are not associated with one another.

In this Specification, the present invention will be described with reference to communication between a node I coupled to private network 22, and a node J coupled to the private network 30, as shown in **Figure 2**. The nodes I and J represent computers, such as the computer illustrated in **Figure 1,** coupled to their respective networks. For simplicity and ease of understanding, an operation by, for example, "node I", shall be understood to mean an operation by the computer coupled to network 22. It will also be noted that although **Figure 2** represents nodes I and J as intermediate or end user computers, that the present invention may also be applied to firewalls. In such event, nodes I and J would represent firewall machines coupled between their respective networks and the Internet 20.

One way to obtain authenticity and privacy at a datagram layer is to use RSA public key certificates. Traditionally, in the event node I desires to send a datagram to, for example, node J, the node I communicates with node J and authenticates itself using a certificate based key management infrastructure. An example of a certificate based infrastructure key management for secure Internet e-mail is the Privacy Enhanced Mail (PEM) system.

The certificates used by PEM are RSA public key certificates. An RSA public key certificate is one which contains an RSA public key. (See, A.Aziz, W.Diffie, "Privacy and Authentication for Wireless LANs", IEEE Personal Communications, February 1994: and also, W.Diffie, M.Wiener, P.Oorschot, "Authentication and Authenticated Key Exchanges".)

There are two primary ways in which RSA certificates can be used to provide authenticity and privacy for a datagram protocol. The first way is to use out-of-band establishment of an authenticated session key, using one of several session key establishment protocols. This session key can then be used to encrypt IP data traffic.

Such a scheme has the disadvantage of establishing and maintaining a pseudo session state on top of a session-less protocol. The IP source must first communicate with the IP destination to acquire this session key. In addition, when the session key must to be changed to insure security, the IP source and the IP destination need to communicate again to effectuate the change. Each such communication involves the use of a computationally expensive public-key operation. This communication requirement is particularly ill-suited to a datagram protocol like IP, which does not require the receiving computer to be in operation to send packets to it, although to establish and change negotiated session keys the receiving computer must be operational.

The second way an RSA certificate can be used to provide authenticity and privacy in a datagram protocol is to complete in-band signaling of the packet encryption key, such that the packet encryption key is encrypted in the recipient's public key. This is the method PEM utilizes to accomplish message encryption. Although this avoids the session state establishment requirement, and also does not require the two parties to communicate to set up and change packet encryption keys, this scheme has the disadvantage of having to carry the packet encryption key encrypted in the recipient's public key in every packet. Since an RSA encrypted key would minimally need to be 64 bytes, and can be 128 bytes, this scheme incurs the overhead of 64-128 bytes of keying information in every packet. In addition, when the packet encryption key changes, a public key operation would need to be performed to recover the new packet encryption key. Thus, both the protocol and computational overhead of such a scheme is high.

The use of Diffie-Hellman (DH) public-key certificates avoids the pseudo session state establishment, and the communications requirement between the two communicating computers to acquire and change packet encrypting keys (see, W. Diffie, M.Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory). Furthermore, the use of a DH public-key certificate does not incur the overhead of carrying 64-128 bytes of keying information in every packet, and is better suited to protocols such as IP, since it does not require the receiving computer to be operational to establish and change packet encrypting keys.

Referring now to the flow charts illustrated in **Figures 3** and **4**, the present invention utilizes DH public-key certificates for key management, such that each IP source and destination is provided with a Diffie-Hellman public key. This DH public-key is distributed in the form of a certificate. The certificate can be signed using either an RSA or DSA signature algorithm. The certificate is referred to herein as a "Diffie-Hellman" (DH) certificate, because the public value that is certified is a Diffie-Hellman public value.

It will be appreciated that the present invention's use of DH certificates to compute a shared key is fundamentally different than the use of the DH certificate to negotiate a session key, for example, as described in the paper by Whitfield Diffie, entitled "Authentication and Authenticated Key Exchanges" (Kluwer Academic Publishers, 1992), because the present invention uses a zero-message protocol to compute a shared secret. All past uses of DH certificates have involved exchanging messages between the two communicating parties.

As will be described, the present invention discloses an improved method and apparatus for generating additional implicit interchange keys for use with the SKIP scheme. For purposes of this Specification, a "context" is an implicit interchange key, where an interchange key is a key which is used to encrypt other keys, as opposed to a traffic key. For example, in the parent applications K_{ij} was considered the interchange key. Accordingly, it is desirable to step the interchange key creating new additional interchange keys, thereby creating further secrets from the original implicit pair-wise interchange key generated. In accordance with the original teachings in the parent applications, upon initialization, each IP source or destination computer, for example node I, is provided with a secret value i, and computes a public value μ^i mod p. Similarly, node J is provided with a secret value j, and computes a public value μ^j mod p. For purposes of illustration, assume that node I wishes to communicate to node J coupled to private network 30 in **Figure 2.** As previously described in the parent applications, both I and J can acquire a shared secret μ^{ij} mod p without having to communicate, so long as the public key of each IP node is known to all other IP nodes. The values μ and p are system parameters, where p is a prime number. It will be appreciated by one skilled in the art that local caching of DH certificates can eliminate the constant need for directory service, thereby minimizing system overhead.

One improvement of the present invention is to create additional shared secrets of μ^{ij} mod p. Utilizing the SKIP scheme, the computable shared secret is used as a key encrypting key to provide for IP packet based authentication and encryption. Thus, μ^{ij} mod p was defined in the original SKIP scheme as a "long term key", and the value K_{ij} was derived from this long term key. The key K_{ij} is used as the key for a known shared key cryptosystem (SKCS) such as DES or RC2. K_{ij} constitutes an *implicit* pair-wise share secret, since K_{ij} does not need to be sent in every packet or negotiated out of band. Simply by examining the source of an IP packet, the destination IP node (for example node J) may compute the shared secret K_{ij}.

As disclosed herein, the successive application of a function to the value μ^{ij} mod p results in the creation of additional implicit pair-wise shared secrets without the necessity of communicating the shared secret between nodes I and J or obtaining a new DH certificate.

In this Specification, the value $[K_{ij}]_N$ denotes the Nth implicit key, where N is indicated in the received packet, as will be described more fully below. Additionally, the quantity $[K_{ij}]_N$ may be determined by evaluating the quantity μ^{Nij} mod p where N = 1, 2, ... or alternatively, $\mu^{(MN)ij}$ mod p; where N = 0.1.2 ... Z. and M = 2,3.4 ... y. As in the original SKIP scheme which is the subject of my parent applications, the key $[K_{ij}]_N$ is derived from the appropriate quantity of μ^{Nij} mod p (or $\mu^{(MN)ij}$ mod p) by using low order key size bits of the respective quantity. The way in which μ^{Nij} is computed is by a simple extrapolation of the DH scheme. Each node computes $(\mu^i)^N = \mu^{iN}$, and in turn raise it to its secret $(\mu^{iN})^j = \mu^{Nij}$. In this Specification, it will be appreciated that the present invention may be realized using either μ^{Nij} mod p, or alternatively, $\mu^{(MN)ij}$. The use of $\mu^{(MN)ij}$ mod p provides additional security, and as will be described, forward secrecy. N is a number that is stored on a pairwise basis in each node. The value of N may be different for different nodes, J, K, ..., etc.

Since μ^{Nij} mod p is minimally at least 512 bytes (and for greater security may be 1024 bytes or higher), sufficient bytes may be derived for use as K_{ij} used as a key for the SKCS. Typically, SKCS key sizes are in the range of 40-172 bits.

As provided by the SKIP scheme, the present invention then utilizes the key $[K_{ij}]_N$ to encrypt a "transient key", which is referred to as K_p. The key K_p is generated at random to encrypt a configurable number of data packets. After the configurable number of data packets have been sent, a new K_p is generated at random. The transient key K_p is used to encrypt an IP data packet, or a collection of IP data packets. The encryption using K_p limits the amount of data in the long-term key which a potential cracker can access. Since it is desirable to retain the long-term key for a relatively long period of time (one or two years), the actual IP data traffic is not encrypted in key $[K_{ij}]_N$. In the preferred embodiment of the invention, only the transient keys of the long-tern key are encrypted using $[K_{ij}]_N$, and the transient keys are used to encrypt IP data traffic. Thus, the amount of data encrypted in the long-term key $[K_{iJ}]_N$ is limited to a relatively small amount over a long period of time.

For purposes of explanation, assume that $[K_{ij}]_N$ is derived from μ^{Nij} mod p. As previously noted, the generalized one-way function of $\mu^{(MN)ij}$ mod p may also be used. The first time the IP source, such as node I, which has been provided with the secret value i, communicates with the IP node J which has been provided with a secret value j, the node I computes the shared secret μ^{Nij} mod p. As will be described, the value of N is an internal value which is initially set by each node to be equal to 1, and incremented based on time and updated using the value of N stored in received packets. In the presently preferred embodiment of the invention, the value N is disposed within a field identified as the Security Association ID ("SAID") in a Specification of IP Security Protocol (IPSP) defined by the Internet Engineering Task Force. The SAID field includes user definable bytes which the present invention utilizes to transmit the value of N. However, it will be appreciated that a variety of mechanisms may be utilized to transmit the value of N, and that the use of the SAID field is only one mechanism of many.

Referring now to **Figure 3,** the sequence of steps utilized by the present invention to encrypt and transmit a packet is illustrated. As shown, a transmitting node I initially sets an internal value of N_i = to 1 (or $N_i = 0$, for the $\mu^{(MN)ij}$ case), and determines whether or not node I has a previously cached $[K_{ij}]_{N_i}$ key. The first time an IP source, such as node I, which has been provided with the secret value i, communicates with the IP node J which has been provided with the secret value j, the node I computes the shared secret μ^{Nij} mod p. To calculate Ut^{Nij} mod p, node I must determine if it has a cached authenticated public DH key for node J. If node I does not have this DH key, it must obtain node J's DH certificate, verify the DH certificate and cache node J's DH public value (see **Figure 3**). The value of $[K_{ij}]_{N_j}$ is then computed and cached.

As illustrated in **Figure 3,** if node I has a cached $[K_{ij}]_{N_i}$ key, and a cached authenticated public DH key for node J, these steps are not repeated. Node I then generates a random key K_p and encrypts this key using $[K_{ij}]_{N_i}$. Node I then encrypts the IP packet data in K_p, and encrypts K_p in $[K_{ij}]_{N_i}$. Node I then transmits the encrypted IP datagram packet and the encrypted key K_p to the receiving node J. Node I further includes its current internal value of N_i in the SAID field of the outgoing packet. The outgoing datagram packet sent by the source node I takes the form illustrated in **Figure 6.**

Referring now to **Figures 4(a), 4(b)** and **5**, the steps for receiving and decrypting the data packet sent by node I to node J will be described. As shown in **Figure 4 (a),** the receiving node J initially sets its internal value of N_j as an variable which is changed internally, or upon the receipt of a valid encrypted package. For example, as shown in **Figure 4(a)** after a predetermined time the internal value of N_j in node J is incremented by N_j+1. The incrementing of N is done in the same manner for each node as that described herein with reference to node J. However, for simplicity and ease of understanding, only the case of node J is described in this Specification.

Upon the receipt of an encrypted data packet from node I the internal value of the variable N_j at node J is compared to the value of N_i in the SAID field of the received packet. If N_i is greater than node J's internal value N_j, node J determines if it has a cached and verified public key for node I. If node J does not have the public key, it obtains a DH certificate for node I and verifies and extracts a DH public value for node I. Node J then computes the value of $[K_{ij}]_{N_i}$ and decrypts the packet (P) using $[K_{ij}]_{N_i}$, and K_p. Node J then determines if the packet P is a valid encrypted packet in the context N_i. This determination may be accomplished in using a number of mechanisms including checksum verifications, examining header formats and the like. If the packet is not validly encrypted in context N_i, then the data

packet is considered invalid and discarded. If the data packet (P) is determined to be a valid encrypted packet in the context Nj then the internal value of N_j for node J is set equal to N_i. Normal data packet processing is then done by node J.

Continuing to refer now to **Figures 5(a)** and **5(b)**, if node J determines that the value of N_i in the SAID field is less than node J's internal value N_j, the data packet is considered to be invalid and discarded. Since in accordance with the teachings with the present invention the value of N_i and N_j may only increase, the case where a received data packet provides a value of N_i which is less than N_j denotes an error condition.

If N_i is equal to N_j, node J determines if it has a cached and verified DH public key for node I. If node J does not have a cached and verified DH public key for node I, node J proceeds to obtain the DH certificate for node I, verifies the certificate, and extracts the DH public value for node I. The value of $[K_{ij}]_{N_j}$ is then determined and cached for later use. As illustrated in **Figure 3,** node J then proceeds to decrypt the encrypted value of K_p in the received data packet using $[K_{ij}]_{N_j}$, and decrypts the IP data utilizing K_p. Node J then completes normal packet processing with the decrypted data packet. Normal packet processing may include the delivery to an appropriate local transport entity, or other outbound interface.

Referring briefly to **Figure 6,** the Message Indicator (MI) is a field that is used to preserve the statelessness of the protocol of the present invention. If a single key is used to encrypt multiple packets, (which is highly desirable since changing the key on a per packet basis constitutes significant computational overhead) then the packets need to be decryptable regardless of lost or out-of-order packets. The MI field serves this purpose. The actual content of the MI field is dependent on the choice of SKCS used for K_p and its operating mode. For example, if K_p refers to a block cipher (e.g. DES) operating in Cipher-Block-Chaining (CBC) mode, then the MI for the first packet encrypted in key K_p is the Initialization Vector (IV). For subsequent packets, the MI is the last blocksize-bits of ciphertext of the last (in transmit order) packet. For DES or RC2 this would be 64 bits. For stream ciphers like RC4, the MI is simply the count of bytes that have already been encrypted in key K_p (and may also be 64 bits).

If the source node I decides to change the packet encryption key K_p, the receiving node J can discover this fact without having to perform a public-key operation. The receiving node J uses the cached value $[K_{ij}]_{N_j}$ to decrypt the encrypted packet key K_p, and this is a shared-key cryptosystem operation. Thus, without requiring communication between transmitting (I) and receiving (J) ends, and without necessitating the use of a public-key operation, the packet encrypting key can be changed by the transmitting side.

Since DH certificates are used, the nodes I and J have no public-key signature algorithm. It will be appreciated that the lack of a public-key signature algorithm is not a major issue, since signing each packet using a public-key cryptosystem is too cumbersome in any case. In accordance with the present invention, the integrity of the packets is determined in a pair-wise fashion using a SKCS.

Application of One-Way Functions

Perfect Forward Secrecy in Closed User Groups

In any security system, it is generally assumed that the keys are not compromised. For sake of example, assume that a cracker successfully obtains K_p. The cracker can also learn the encryption of K_p under K_{ij} because this information is part of the packet header. The cracker may then send forged traffic to either nodes I or J pretending to be the other node. In addition, if a cracker learns a node's secret (e.g., i), this allows the cracker to compute $K_{ij} = \mu^{ij} \bmod p$, and thus to decrypt all traffic that was encrypted by using K_{ij} as the interchange key.

The protocols described the preceding portions of this specification can be employed in a special situation, the closed user group, to limit the damage that would otherwise occur if an interchange key $[X_{ij}]_N$ is compromised. A closed user group is one whose membership is determined once and for all at a given point in time: no new members are added after that point. The damage limitation feature of the invention, explained below, is referred to as "perfect forward secrecy."

In accordance with the teachings of the preceding portions of this specification, the nodes of the closed user group will periodically change the interchange key $[K_{ij}]_N$ which they use. The key in use at any particular point in time will be the Nth level key, $[K_{ij}]_N$, for some integer N. Periodically, the nodes of the closed user group will increment N, and thus begin using a new interchange key $[K_{ij}]_{N+1}$.

The term "perfect forward secrecy" is used in this specification to mean that if an Nth level key $[K_{ij}]_N$ is compromised, an intruder will not be able to read traffic encrypted with previous keys $[K_{ij}]_S$ for $S < N$. Through an appropriate choice of the function f(z) which is used to obtain the interchange key $[K_{ij}]_N$ from the previous interchange key $[K_{ij}]_{N-1}$, and with appropriate choice of the modulus p, perfect forward secrecy may be achieved. The key to perfect forward secrecy is that f be a one-way function, that is to say, one which is difficult to invert, so that if one knows f(z), it is difficult to compute z.

An appropriate choice of the function f(z) is exponentiation by a number $M > 2$ with respect to a modulus p, so that $f(z) = z^M \bmod p$. If the first interchange key $[K_{ij}]_0 = \mu^{ij} \bmod p$, the second key will be $f([K_{ij}]_0) = (\mu^{ij})^M \bmod p = \mu^{ijM} \bmod$

p, the third $f([K_{ij}]_1) = (\mu^{ijM})^M \bmod p = \mu^{ijM^2} \bmod p$, and so forth, with $[K_{ij}]_N = \mu^{ijM^N} \bmod p$ for all N. With this choice of the function f(z), in order to compute $[K_{ij}]_N$ given $[K_{ij}]_S$ with S < N, one simply raises $[K_{ij}]_S$ to the (M^{N-S})th power:

$$(\mu^{ijM^S})^{M^{N-S}} \bmod p = (\mu^{ijM^{S+N-S}}) \bmod p = (\mu^{ijM^N}) \bmod p.$$

In contrast, in order to compute $[K_{ij}]_{N-1}$ given $[K_{ij}]_N$, it is necessary to take the Mth root of $[K_{ij}]_N$ modulo p.

Efficient algorithms for computing Mth roots modulo p are known for p prime. One such algorithm is given in Henri Cohen, *A Course in Computational Algebraic Number Theory* § 1.6 (1993). These algorithms can be extended to composite p whose factorization into primes is known. However, no such efficient algorithm is known for computing Mth roots modulo p if p is a composite number whose factorization into primes is unknown. If there were such an efficient algorithm, then the widely used Rivest-Shamir-Adleman (RSA) cryptosystem would be insecure: the fact that people have been trying to crack RSA for almost 20 years provides some assurance that Mth roots modulo a composite number are hard to compute. For this reason, if $f(z) = z^M \bmod p$ is chosen as the function for going from the Nth to the (N+1)st interchange key, and p is a composite number which is difficult to factor, then f(z) is a one-way function.

For exponentiation modulo p to be a true one-way function, it is important that p be chosen to be a composite number which is difficult to factor. A way of choosing such a p is to pick two large prime numbers p_1 and p_2 and to set $p = p_1 p_2$. More generally, there is considerable experience in choosing large numbers which are difficult to factor for use in conjunction with the RSA cryptographic algorithm, whose security depends on the computational difficulty of factoring.

A further action which is important for ensuring perfect forward secrecy is for each node I to delete its own DH secret i after computing the initial set of shared keys $K_{ij} = \mu^{ij} \bmod p$ for all other nodes J (using those nodes' respective DH public keys $\mu^j \bmod p$). This deletion of the DH secret i is valuable because, in practice, it is likely that the compromise of an interchange key $[K_{ij}]_N$ will occur (if it occurs at all) because a cracker has managed to obtain access to the entire contents of some node I, as for example by obtaining the password of a user account with complete control over the node. When the cracker has complete control of a node I, and the DH secret i is still on the node, the cracker will be able to compute the initial interchange keys $[K_{ij}]_I = \mu^{ij} \bmod p$ and thus all subsequent interchange keys $[K_{ij}]_N$. This unfortunate possibility can be avoided by the simple expedient of deleting i once it is no longer needed.

Note that if node I deletes its DH secret i, it is no longer possible to add a new member L with a new secret *l* to the group in the normal manner, because node I will be unable to compute the pairwise interchange key $K_{il} = \mu^{il} \bmod p$ just from a knowledge of node L's public key $\mu^l \bmod p$. This is why the perfect forward secrecy scheme just described is suitable primarily for closed user groups to which new modes L are not added.

For similar reasons, once a node I determines that communication with the other nodes in the user group is proceeding in terms of a particular $[K_{ij}]_N$, so that no further traffic with $[K_{ij}]_S$ for S < N will need to be read or generated, then the node should delete any copies it has of $[K_{ij}]_S$ for any S < N after a timeout period. The decision that communication with the other nodes in the user group is employing a particular $[K_{ij}]_N$ is made as described previously in the specification. That is to say, upon the receipt of a data packet (P) in the context N_i (e.g., a data packet from node I), node J computes $[K_{ij}]_{N_i}$ and decrypts the data packet P. Node J then determines whether the data packet was a valid encrypted packet in context N_i using one of a number of mechanisms previously described with reference to **Figures 4** and **5**. If the packet P is a valid packet in context N_i, node J sets its internal context $N_j = N_i$ and concludes that no further packets are being originated with contexts $S < N_i$, so that the old contexts can be discarded after a period of time which represents the maximum packet transmission delay in the network. This is particularly important for the application of this technique to protocols such as IP which do not guarantee that the packets arrive in order.

If exponentiation is being used as the one-way function f(z), and a composite modulus p is being employed, it is important to note that the modulus p must be obtained from a source which can be trusted to provide a modulus which is difficult to factor. A cracker who managed to give a closed user group a p of the cracker's own choosing (for example, a prime p, or a p whose factorization into primes the cracker knows) would be able to invert the function $f(z) = z^M \bmod p$, thus defeating the perfect forward secrecy function described in the preceding paragraphs. Furthermore, if the trusted party generated p by multiplying random primes, it is desirable that the trusted party forget those primes after having communicated their product to the nodes in the closed user group, lest a cracker penetrate the trusted system and be able to discover factorizations which have remained lying around. This is a difference between the generation of moduli for this system as compared to RSA, in which it is expected that one of the communicating parties remembers the factorization for as long as a key pair is in use.

As will be appreciated, for closed user groups where each node may precompute all secret values and then discard their secrets, perfect forward secrecy may be obtained using one-way functions f(z). In the event a new node is added to the closed group, each of the nodes must have a new set of DH certificates/secrets assigned to them. However, for long term closed groups, the present invention's application of one way functions and the deletion of each node's DH secret value significantly enhances network security.

It will also be noted that the key management scheme described herein may also be used to provide an integrity check (in addition to secrecy) for packets. In this case, the key K_p may be used directly by either node I or node J to compute a Message Authentication Code (MAC) either over the entire packet or over the portions that require authentication.

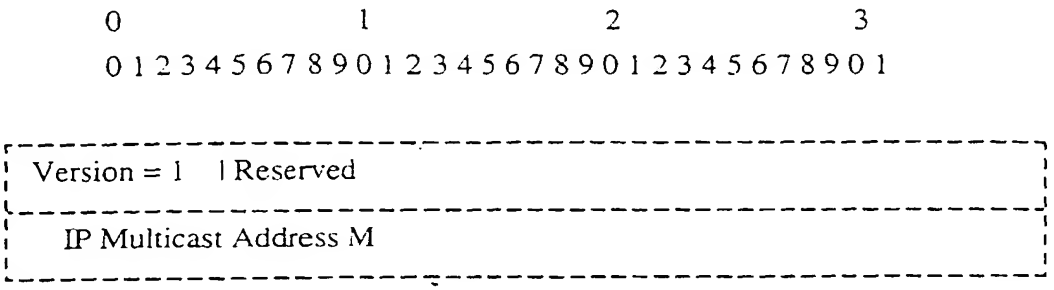
Improved Application of SKIP to Datagram Multicast Protocols

The method of the present invention may be used in conjunction with datagram multicasting protocols such as IP (or IPng) multicast. This application requires key-management awareness in the establishment and joining process of multicast groups. Furthermore, in order to distribute multicast keying material, the notion of a group owner should exist. When secure multicasting to a multicast address M is required, a group membership creation primitive will establish the group key K_g and the membership list of addresses that are allowed to transmit and receive encrypted multicast datagrams to and from group address M. This action will be taken by the group owner.

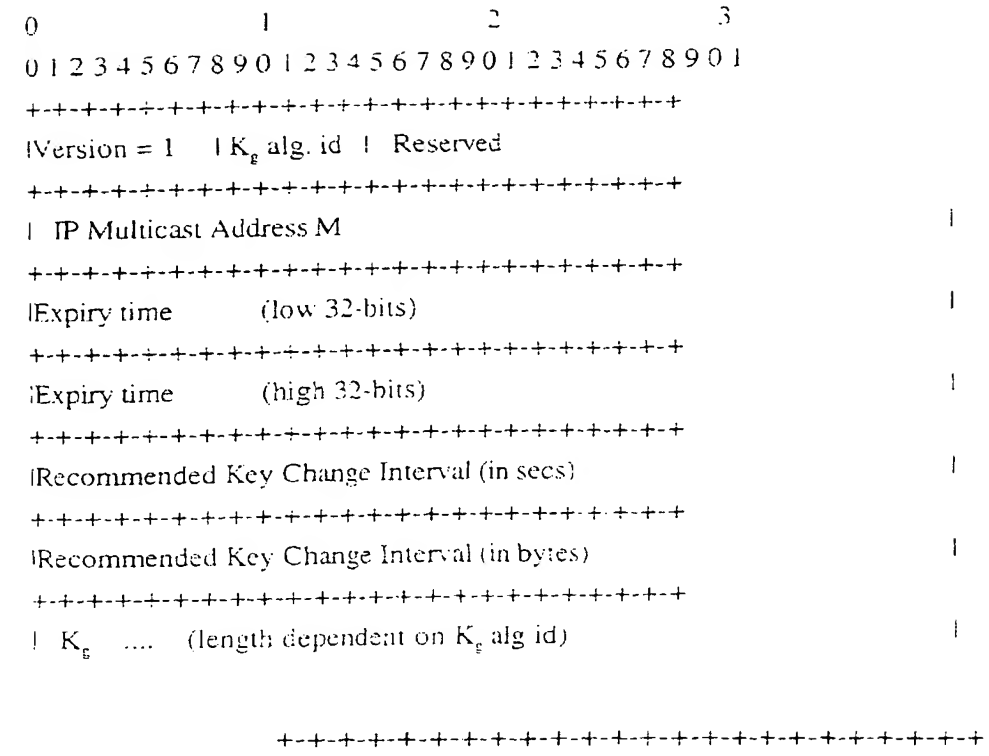
The group key K_g is not used as a packet encryption key, but rather as the Group Interchange Key (GIK). Namely, K_g is used as a key-encrypting-key, similar to way the pair keys $[K_{ij}]_N$ are used in SKIP for unicast IP.

Nodes wishing to transmit/receive encrypted datagrams to multicast address M acquire the GIK K_g. In the present invention, this is accomplished by sending an encrypted/authenticated request-to-join primitive to the group owner. If the requesting node's address is part of the group's authorized membership list, the group owner sends the GIK K_g, algorithm identifier, associated lifetime information and key-change policy in an encrypted packet, using the pair-wise secure protocol previously described in this Specification.

The packet formats for the GIK request/response is given below. This describes the payload portion of either a TCP or UDP packet, which has been enhanced using SKIP unicast procedures. If using UDP, multiple requests may be sent, in case of packet losses of earlier requests/response messages. The request is sent to TCP/UDP port # XXXX corresponding to the group owner's unicast IP address.

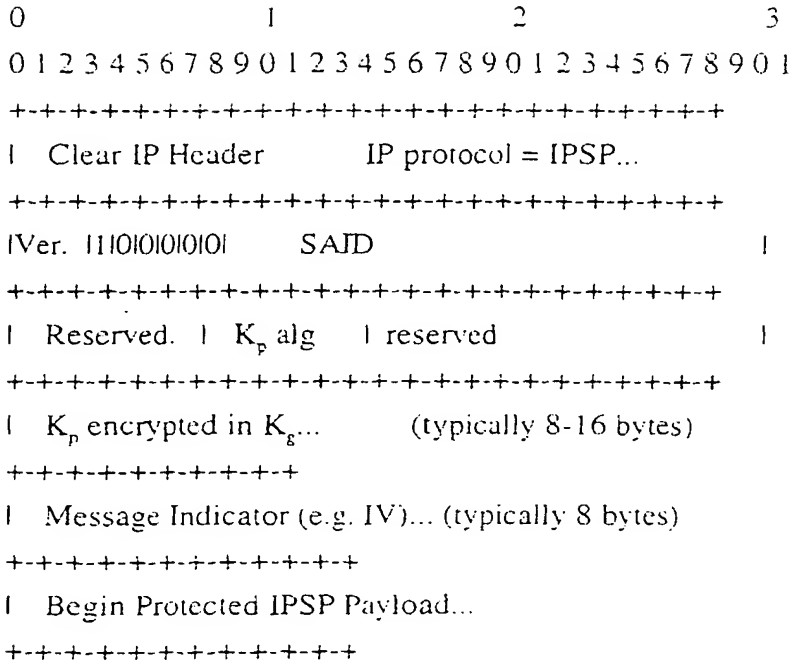


The first field specifies the version of this protocol, which is 1. Following this field is the actual IP multicast address for which the GIK is being requested. The request packet that is sent must have the minimal IPSP enhancement of source-origin authentication, and may optionally be encrypted and/or have playback protection by use of the sequence number field. The group owner's response is an encrypted packet containing the GIK K_g. The response is sent to TCP/UDP port # XXXX and is addressed to the requestor's unicast IP address. This packet format is as follows. As before, it defines the data-portion of a TCP or UDP packet.



The 64-bit expiry time specifies when the multicast key is considered to have expired. This is in terms of seconds since, Jan. 1, 1994, expressed in GMT. The recommended key-change interval is what every source of encrypted traffic to the multicast group uses to determine the key-change policy. There are two ways currently envisioned to specify a key-change policy. The first is in terms of elapsed time since last key-change. Another is in terms of the amount of data encrypted in a given packet encryption key. It is contemplated that each source will use whichever of these methods which determines the more frequent key-change policy, whether this is in terms of amount of traffic encrypted in a given key, or in terms of elapsed time (in seconds) since the last key change.

Transmitting nodes to group address M will randomly generate packet encryption keys K_p, and encrypt these keys using K_g. The packet structure is similar to the structure used for encrypted unicast SKIP packets, except that the packet keys K_p are not encrypted in the pair-wise keys K_{ij}, but instead are encrypted using the GIK K_g. An example encrypted multicast packet is shown below.



The destination IP address will be used by the receiver to determine whether to use unicast of multicast key-processing procedures on a received IP packet. In case the destination address is an IP multicast address, it will use the group IK K_g to decrypt the packet encryption key K_p.

There are two distinct advantages of this scheme. Every member of the multicast group can change packet encryption keys as often as required (in line with the policy set by the group owner), without involving key-setup communications overhead involving every member of the group. This can be extremely frequent. even once every few seconds, even with very large multicast groups, because there is no extra communications overhead for changing packet encryption keys.

In addition, since all the packet encryption keys are randomly generated, and hence different, there is no problem in using stream-ciphers with multicast. This is because each source of encrypted traffic when using a stream cipher would use a different key-stream and thus there is no key-stream reuse problem. If all members of the multicast group used the same packet encryption key, then certain stream ciphers could not be used with multicast IP.

An implementation of this improved protocol will use the destination IP multicast address to look-up the GIK K_g.

How the identity of the group owner is established and communicated to the participating nodes is left to the application layer. However, it will be appreciated that this should be done in a secure fashion, otherwise the underlying key-management facility may be defeated.

An advantage of the method of the present invention is that only the keying information is distributed in a pair-wise fashion. The actual encrypted data packet is sent using the standard multicast delivery mechanisms, thereby allowing the same network bandwidth efficiency that is expected of a network layer multicast protocol when operating over subnetworks which also support multicasting (for example, Ethernet, FDDI, etc). This scheme is considered to scale well, even for a large number of nodes, because key-change requires no extra communications overhead.

Management of DH Certificates

Since the nodes' public DH values are communicated in the form of certificates, the same type of multi-tier certification structure that is being deployed for PEM, and also by the European PASSWORD. There may be a Top Level Certifying Authority (TLCA) which may constitute the same the Internet Policy Registration Authority (IPRA), Policy Certifying Authorities (PCAs) at the second tier and the organizational Certificate Authorities (CAs) below that.

In addition to the identity certificates, which are part of PEM, additional authorization certificates are needed to properly track the ownership of IP addresses. Since it is desirable to directly use IP addresses in the DH certificates, name subordination principles alone cannot be used to determine if a particular CA has the authority to bind a particular IP address to a DH public key. However, the present invention may use the X.509/PEM certificate format, since the

subject Distinguished Name (DN) in the certificate can be the ASCII decimal representation of an IP (or IPng) address.

Since the nodes only have DH public keys, which have no signature capability, the nodes themselves are unable to issue DH certificates. The node certificates are issued by organizational CAs which have jurisdiction over the range of IP addresses that are being certified. The PCAs will have to perform suitable checks (in line with the policy of that PCA), to confirm that the organization which has jurisdiction over a range of addresses is issued a certificate giving it the authority to certify the DH values of individual nodes with those addresses. This authority may be delegated in the form of a authorization certificate signed by the PCA. For the purposes of authorization, the CA's Distinguished Name (DN) will be bound to the range of IP addresses over which it has jurisdiction. The CA has either an RSA or DSA certificate from the PCA. The CA which has authority over a range of IP addresses can delegate authority over part of the range to a subordinate CA, by signing another authorization certificate using its own private key. The organizational CA so authorized is identified by the range of addresses that it can issue certificates for. The range of IP addresses are identified in the certificate in the form of a IP address prefix length list.

Claims

1. A method for sending data from a first data processing device to a second data processing device in a closed user group having a plurality of nodes, the first data processing device constituting a first node of the plurality of nodes and the second data processing device constituting a second node of the plurality of nodes, the method comprising the steps of:

   providing a unique secret value and a public value to each of said nodes;
   providing each of said nodes with an associated context variable;
   for each of the nodes, obtaining a certificate for each of the other nodes in the plurality of nodes and determining said public values for each of said other nodes from said certificates;
   each of said nodes precomputing a shared secret for each of said other nodes;
   each of said nodes deleting its own unique secret value;
   said first node computing an encryption key from said precomputed shared secret for said first and second nodes;
   encrypting data to be transmitted to the second node using said encryption key;
   said first node sending said encrypted data using said encryption key to said second node;
   said first node notifying said second node of the current value of said first node's associated context variable.

2. The method as defined by claim 1, further comprising the steps by said second node of:

   receiving said data from the first node;
   comparing the second node's associated context variable to the first node's context variable; and
   if the first context variable is greater than the second context variable, computing said key from said precomputed shared secret for said first and second nodes, and setting said second context variable to equal the first context variable if said data is valid.

3. The method as defined by claim 1 or 2, wherein said data to be transmitted to the second node includes a transient key employed to encrypt other data to be transmitted to the second node.

4. The method as defined by claim 3, further including the steps of:

   decrypting said transient key using said encryption key, and
   decrypting said other data with said transient key,

   whereby the second node decrypts data received and previously encrypted by the first node.

5. The method as defined by any of claims 2-4, further comprising the step by said second node of:

   if the first context variable for the first node equals the second context variable for the second node, computing said encryption key from said precomputed shared secret for said first and second nodes.

6. The method as defined by any of claims 2-5, further comprising the step by said second node of:

if the second context variable of the second node is greater than the first context variable of the first node, discarding said data and denoting an error condition.

7. The method as defined by any of claims 2-6, further comprising the step, performed after a predetermined time period by each second node, of incrementing the value of said second context variable provided for the second node.

8. The method as defined by claim 7, further comprising the steps, performed after said step of incrementing the value of the second context variable by each second node, of:

computing a next shared secret from a current shared secret for said other nodes, and
deleting said current shared secret for said other nodes.

9. The method as recited in any of the preceding claims, wherein for each second node of said nodes, said second context variable provided for the second node is initially set equal to 1.

10. The method as defined by claim 8 or 9, wherein said next shared secret is computed from said current shared secret by applying a one-way function to said current shared secret.

11. The method as defined by claim 10, wherein said one-way function is equal to $z^M \bmod p$.

12. The method as defined by any of claims 1-9, wherein the encryption key is an implicit pair wise secret used as a key for a shared key cryptosystem (SKCS) and derived from $a^{(MN)ij}$.

13. The method as defined by claim 12, wherein a and p are system parameters, and wherein p is a composite number which is difficult to factor.

14. The method as defined by claim 13, wherein said other data comprises a data packet which includes a source address, a destination address and an SKCS identifier field.

15. The method as defined by claim 12, 13, or 14, wherein said data packet further includes a message indicator field.

16. An apparatus for encrypting data for transmission from a first data processing device to a second data processing device in a closed user group having a plurality of nodes, the first data processing device constituting a first node of the plurality of nodes and the second data processing device constituting a second node of the plurality of nodes, comprising:

each of said nodes including a storage device configured to store a unique secret value, a public value, and an associated context variable:
each of said nodes including a computation device configured to precompute a shared secret for each of said other nodes, using a certificate for each of said other nodes to determine said public values for said other nodes:
the first node including an encrypting device configured to encrypt data to be transmitted to the second node, said encrypting device deriving a encryption key from the precomputed shared secret for said first and second nodes, and
said encrypting device encrypting said data using said encryption key;
the first node further including an interface circuit configured to transmit said encrypted data to said second node and to notify said second node of the value of the first node's associated internal context variable.

17. The apparatus as defined by claim 16, wherein each of said encryption devices for said nodes deletes its own unique secret value after precomputing all of said shared secrets for said other nodes.

18. The apparatus as defined by claim 16 or 17, wherein said node includes:

a receiver for receiving said encrypted data from the first node:
a decrypting device coupled to said receiver for decrypting said encrypted data.

19. The apparatus as defined by any of claims 16-18, wherein said data to be transmitted to the second node comprises a transient key employed to encrypt other data to be transmitted to the second node.

20. The apparatus as defined by any of claims 16-19, wherein said decrypting device comprises:

a comparator configured to compare the first node's associated internal context variable to the second node's associated internal context variable: and

a key-computation device configured to compute said encryption key from said precomputed shared secret for said first and second nodes if the first internal context variable is greater than the second internal context variable:

wherein said decrypting device utilizes said encryption key to decrypt said transient key, decrypts said other data using said transient key, and sets the second internal context variable equal to the first internal context variable.

21. The apparatus as defined by claim 20, wherein if the first internal context variable is equal to the second internal context variable said second node, said second node computes said encryption key from said precomputed shared secret for said first and second nodes, and wherein said encrypting device utilizes said encryption key to decrypt said transient key, and decrypts said other data using said transient key.

22. The apparatus as defined by claim 20 or 21, wherein if the first internal context variable is less than the second internal context variable, said second node discards said data.

23. The apparatus as defined by any of claims 16-22, wherein said first and second nodes increment said first internal context variable and said second internal context variable, respectively after a predetermined time period.

24. The apparatus as defined by claim 23, wherein said second node further computes a next shared secret from a current shared secret, and deletes said current shared secret for all of said other nodes.

25. The apparatus as defined by any of claims 19-24, wherein said other data comprises a data packet which includes a context field for storing the first internal context variable.

26. The apparatus as defined by any of claims 16-25, wherein for each of said nodes, the first and second internal context variables are initially set equal to 1.

27. The apparatus as defined by any of claims 16-26, wherein said public value for each of said nodes is $a^i \bmod p$, where i is said unique secret value for each of said nodes.

28. The apparatus as defined by any of claims 24-27, wherein said next shared secret is computed from said current shared secret by applying a one-way function to said current shared secret.

29. The apparatus as defined by any of claims 16-28, wherein said encryption key is an implicit pair wise secret used as a key for a shared key cryptosystem (SKCS).

30. The apparatus as defined by any of claims 27-29, wherein a and p are system parameters, and where p is a composite number which is difficult to factor.

31. The apparatus as defined by any of claims 25-30, wherein said data packet includes a source address, a destination address and an SKCS identifier field.

32. The apparatus as defined by claim 31, wherein said data packet further includes a message indicator field.

FIG. 1

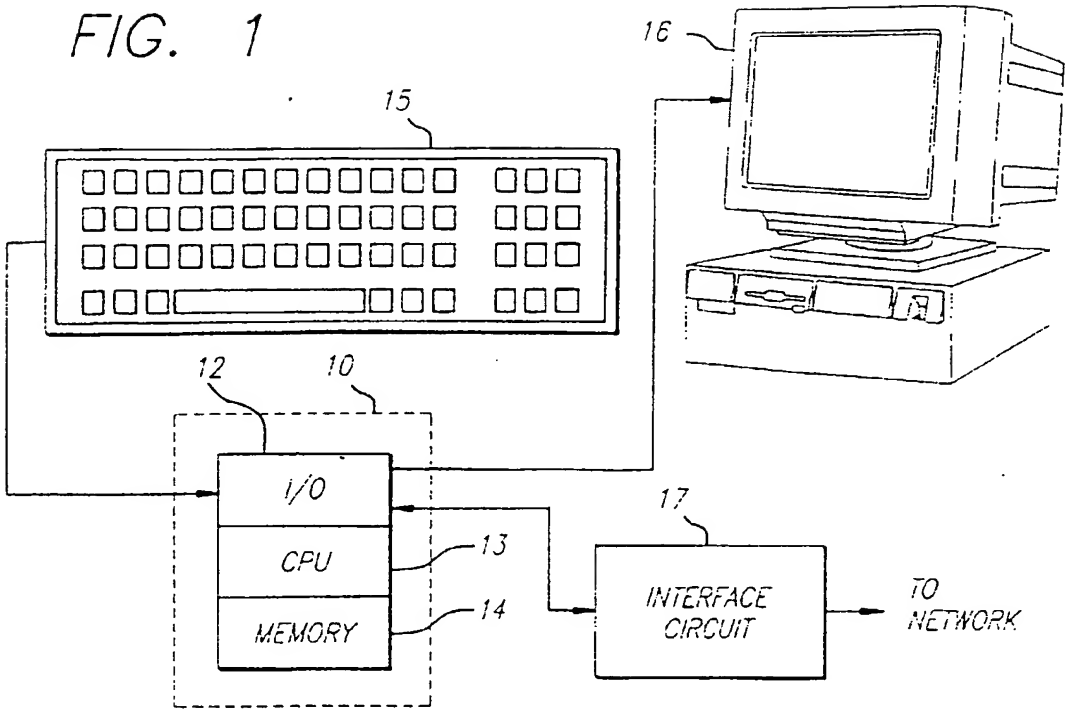

16
15
12
10
I/O
CPU
MEMORY
13
14
17
INTERFACE CIRCUIT
TO NETWORK

FIG. 2

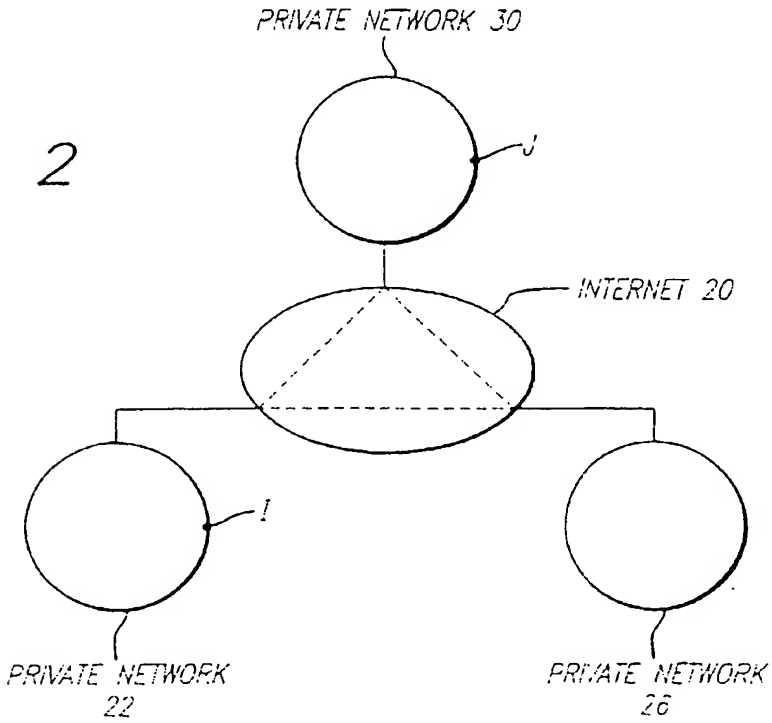
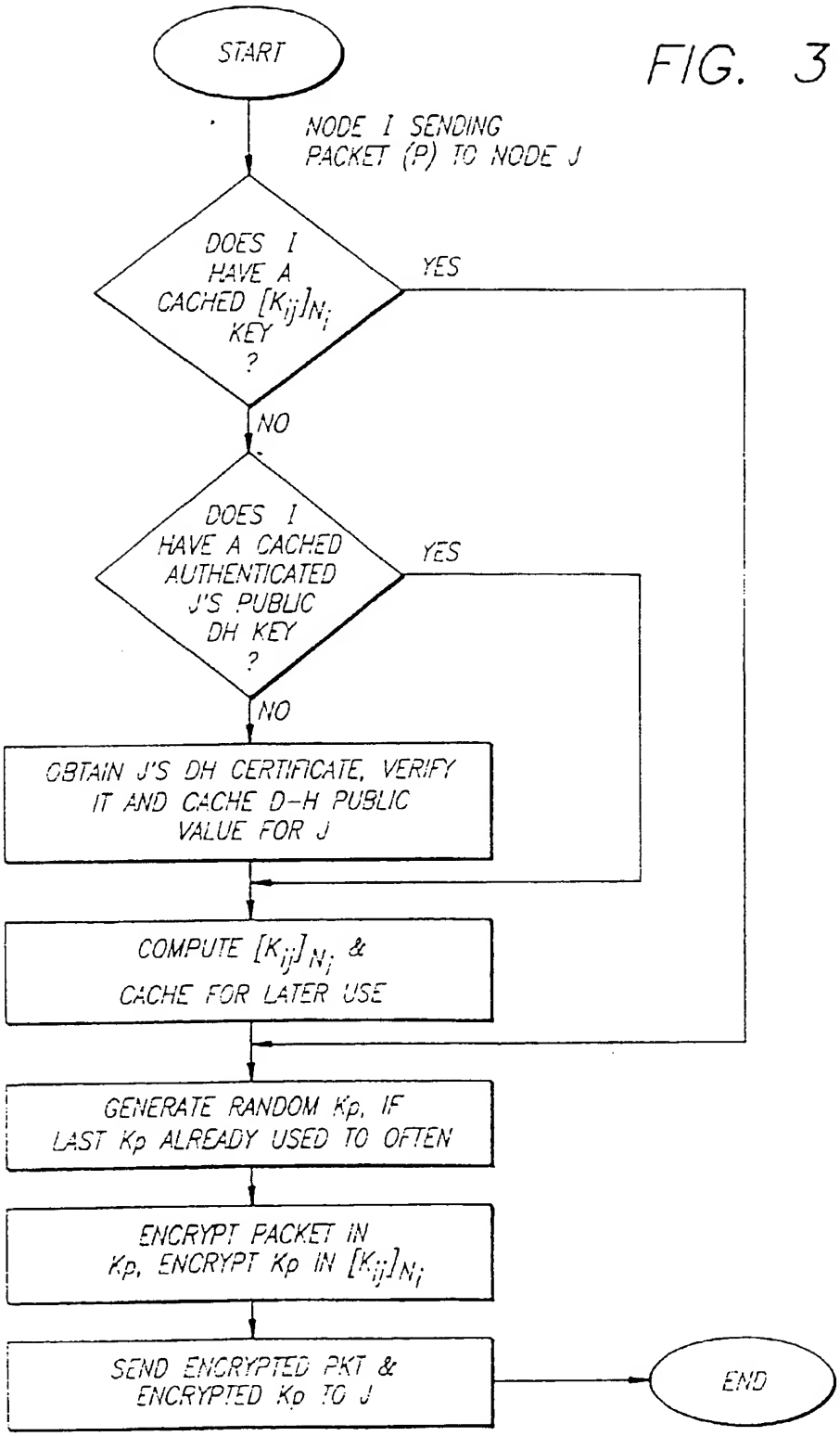

PRIVATE NETWORK 30
J
INTERNET 20
I
PRIVATE NETWORK 22
PRIVATE NETWORK 26

FIG. 3


START
NODE I SENDING PACKET (P) TO NODE J
DOES I HAVE A CACHED $[K_{ij}]_{N_i}$ KEY ?
YES
NO
DOES I HAVE A CACHED AUTHENTICATED J'S PUBLIC DH KEY ?
YES
NO
OBTAIN J'S DH CERTIFICATE, VERIFY IT AND CACHE D-H PUBLIC VALUE FOR J
COMPUTE $[K_{ij}]_{N_i}$ & CACHE FOR LATER USE
GENERATE RANDOM Kp, IF LAST Kp ALREADY USED TO OFTEN
ENCRYPT PACKET IN Kp, ENCRYPT Kp IN $[K_{ij}]_{N_i}$
SEND ENCRYPTED PKT & ENCRYPTED Kp TO J
END

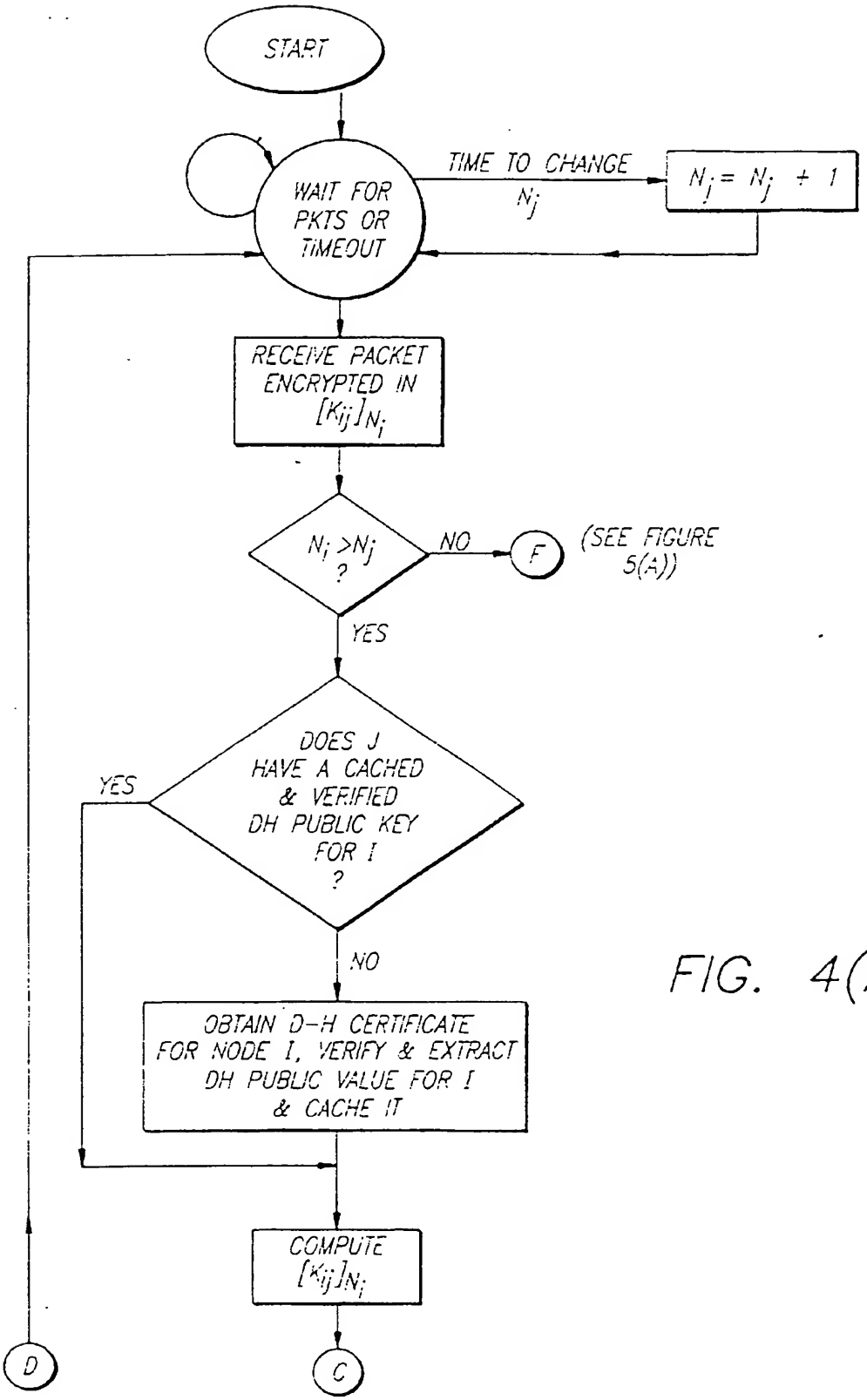

START
WAIT FOR PKTS OR TIMEOUT
TIME TO CHANGE N_j
$N_j = N_j + 1$
RECEIVE PACKET ENCRYPTED IN $[K_{ij}]_{N_i}$
$N_i > N_j$?
NO
F
(SEE FIGURE 5(A))
YES
DOES J HAVE A CACHED & VERIFIED DH PUBLIC KEY FOR I ?
YES
NO
OBTAIN D-H CERTIFICATE FOR NODE I, VERIFY & EXTRACT DH PUBLIC VALUE FOR I & CACHE IT
COMPUTE $[K_{ij}]_{N_i}$
D
C

FIG. 4(A)

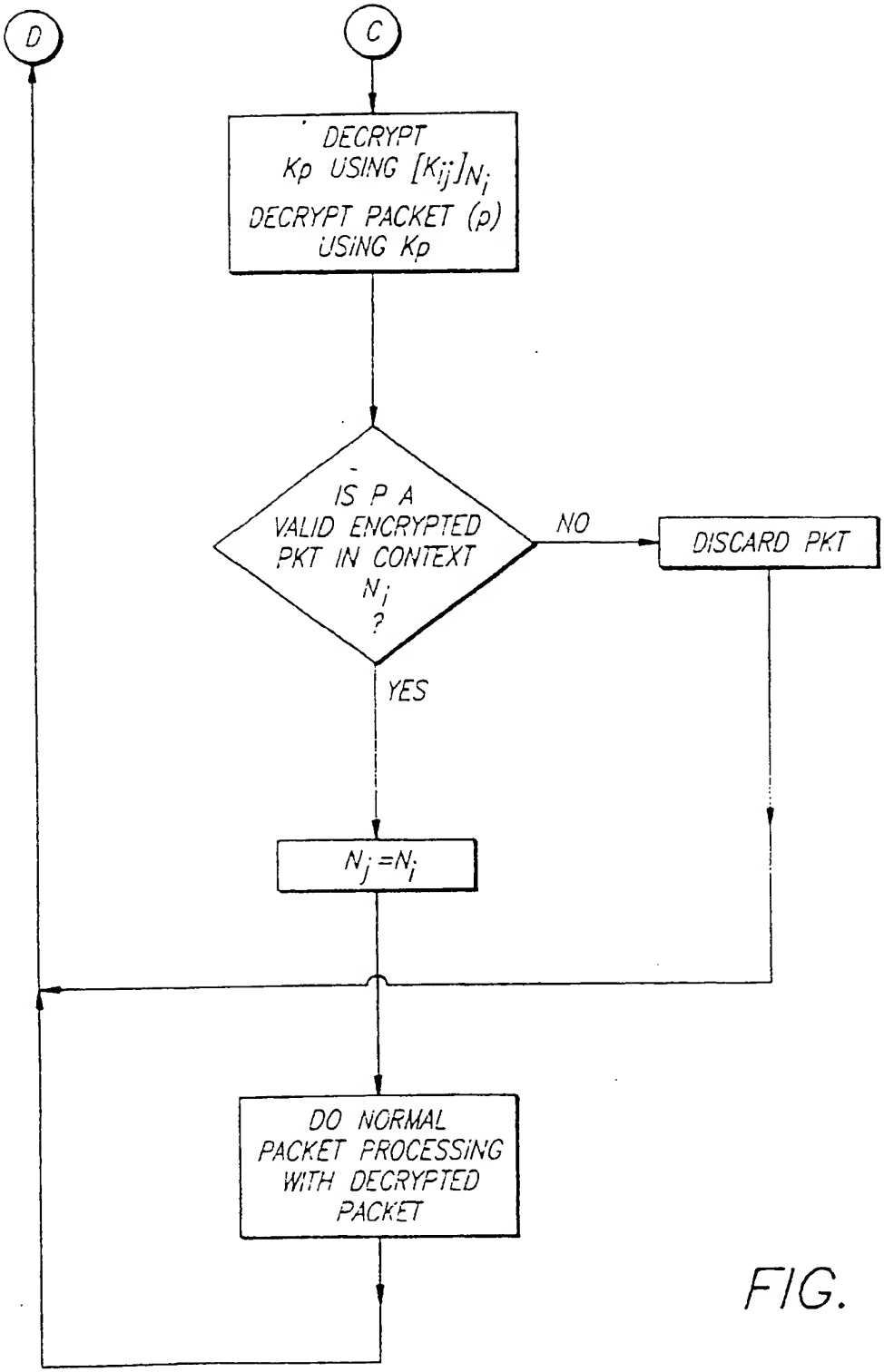

D
C
DECRYPT Kp USING $[K_{ij}]_{N_i}$
DECRYPT PACKET (p) USING Kp
IS P A VALID ENCRYPTED PKT IN CONTEXT N_i ?
NO
DISCARD PKT
YES
$N_j = N_i$
DO NORMAL PACKET PROCESSING WITH DECRYPTED PACKET

FIG. 4(B)

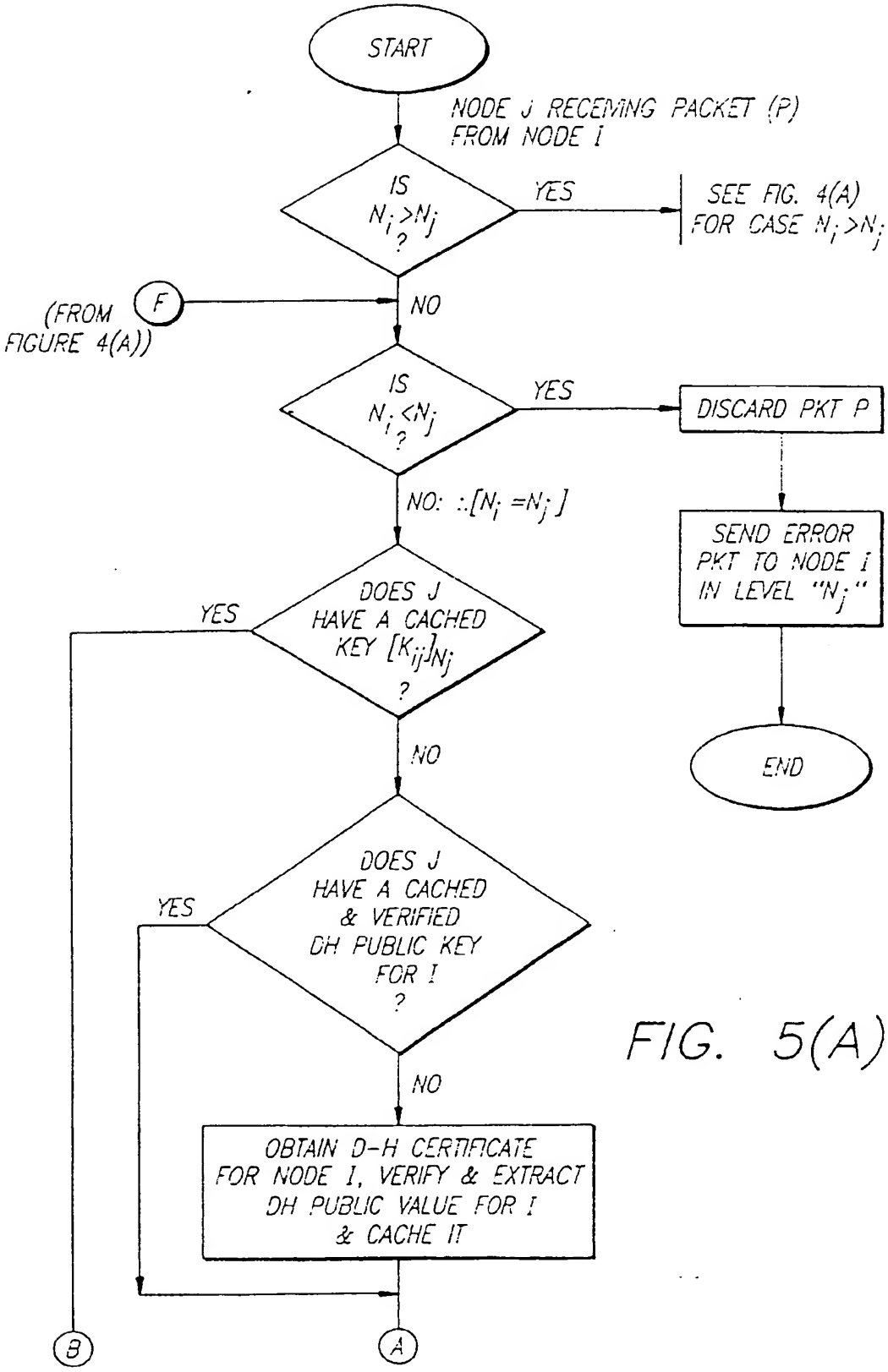

START
NODE J RECEIVING PACKET (P) FROM NODE I
IS $N_i > N_j$?
YES
SEE FIG. 4(A) FOR CASE $N_i > N_j$
(FROM FIGURE 4(A))
F
NO
IS $N_i < N_j$?
YES
DISCARD PKT P
SEND ERROR PKT TO NODE I IN LEVEL "N_j"
END
NO: $\therefore [N_i = N_j]$
DOES J HAVE A CACHED KEY $[K_{ij}]_{Nj}$?
YES
NO
DOES J HAVE A CACHED & VERIFIED DH PUBLIC KEY FOR I ?
YES
NO
OBTAIN D-H CERTIFICATE FOR NODE I, VERIFY & EXTRACT DH PUBLIC VALUE FOR I & CACHE IT
B
A

FIG. 5(A)

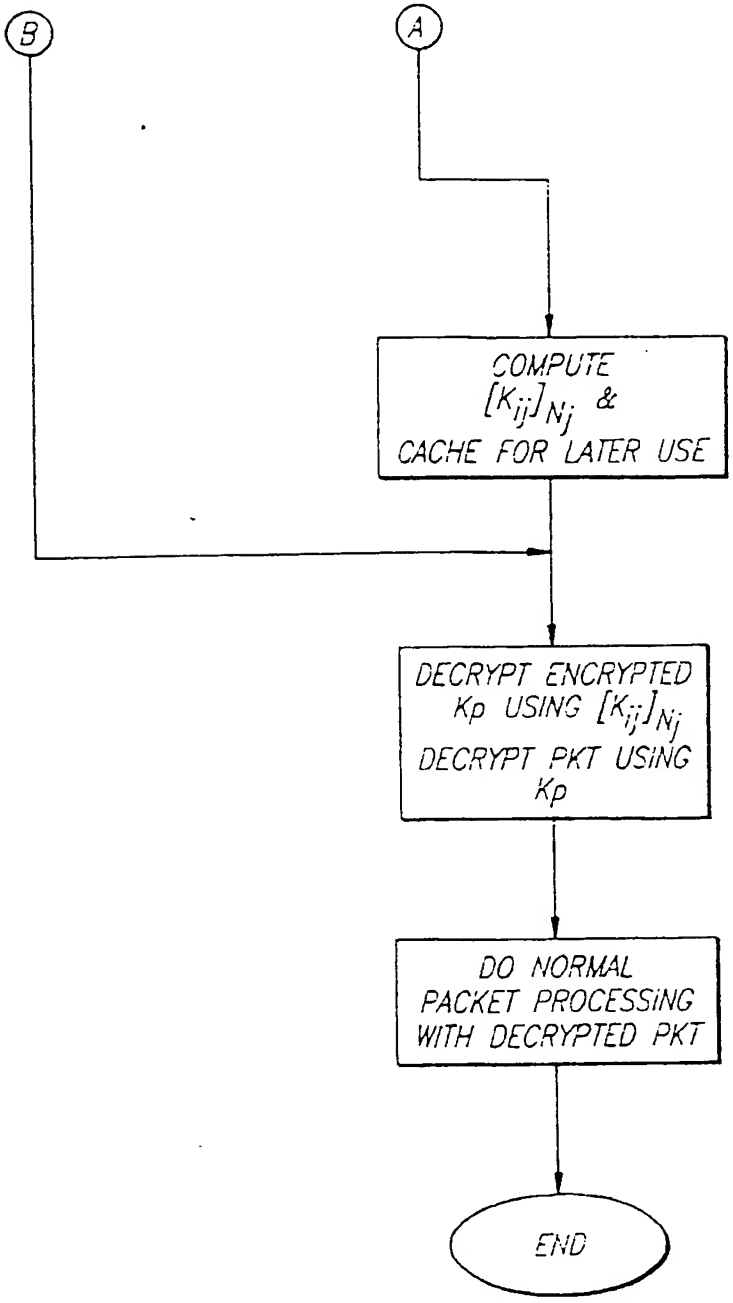

B
A
COMPUTE
$[K_{ij}]_{Nj}$ &
CACHE FOR LATER USE
DECRYPT ENCRYPTED
Kp USING $[K_{ij}]_{Nj}$
DECRYPT PKT USING
Kp
DO NORMAL
PACKET PROCESSING
WITH DECRYPTED PKT
END

FIG. 5(B)

FIG. 6

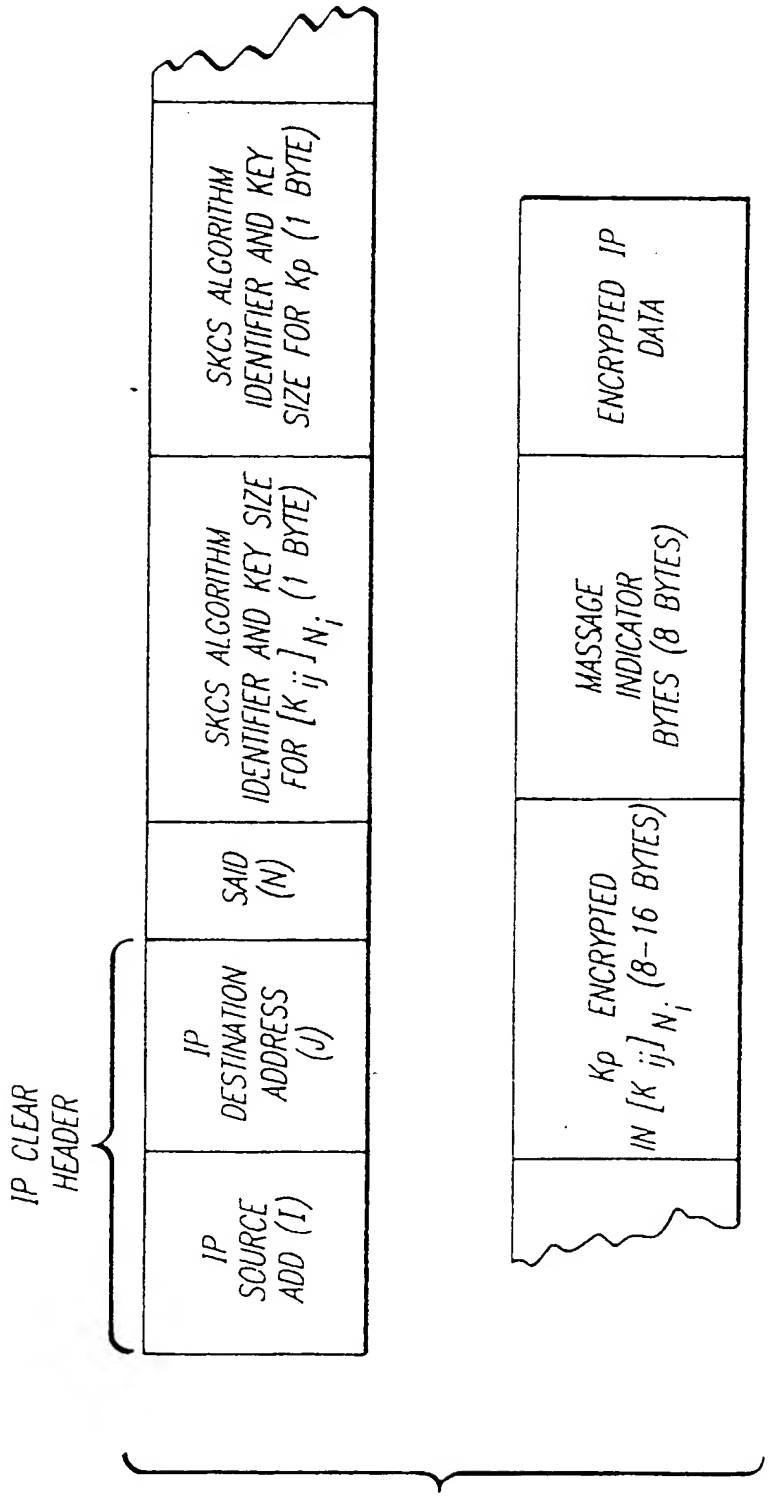
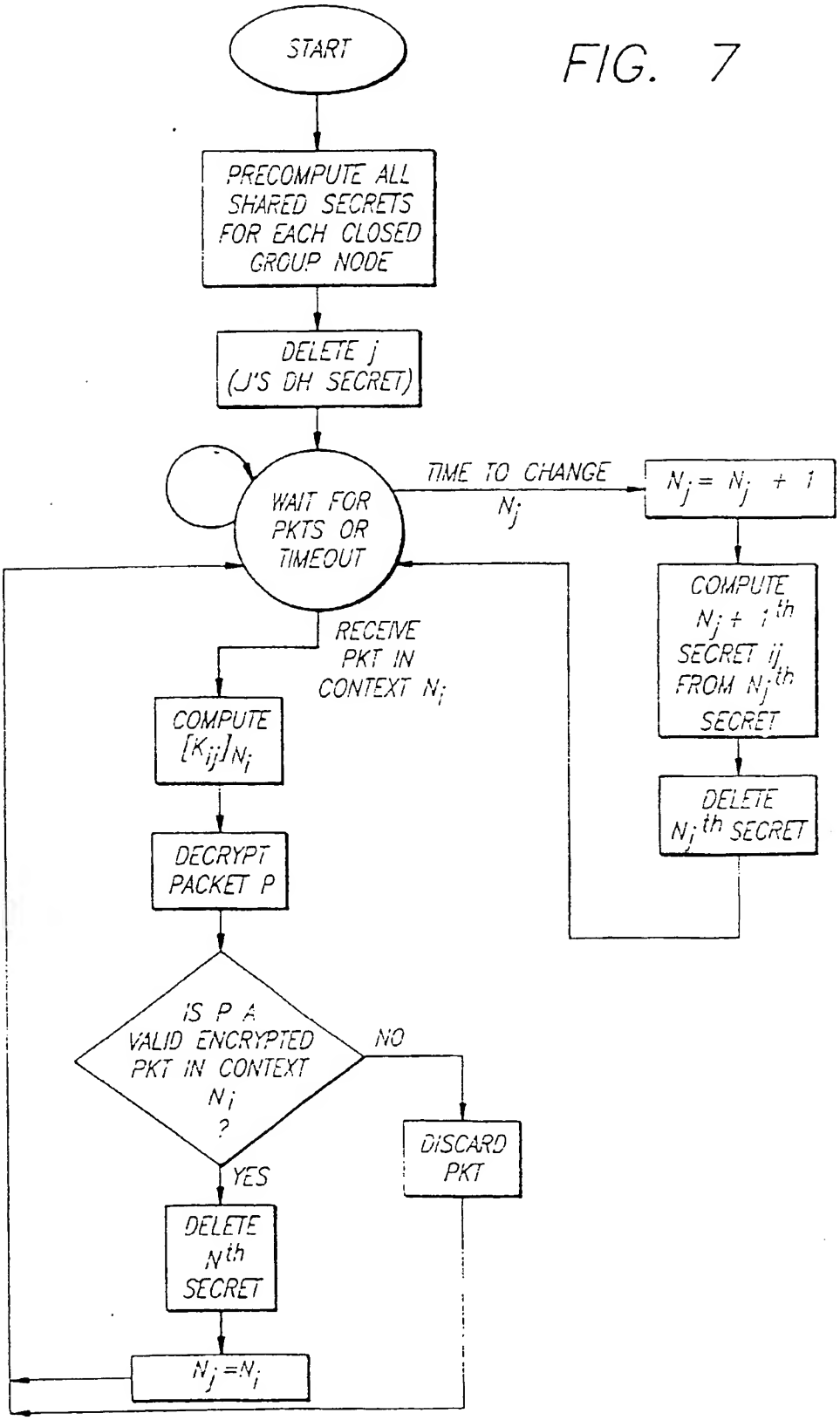

IP CLEAR HEADER
IP SOURCE ADD (I)
IP DESTINATION ADDRESS (J)
SAID (N)
SKCS ALGORITHM IDENTIFIER AND KEY SIZE FOR $[K_{ij}]_{N_i}$ (1 BYTE)
SKCS ALGORITHM IDENTIFIER AND KEY SIZE FOR Kp (1 BYTE)
Kp ENCRYPTED IN $[K_{ij}]_{N_i}$ (8-16 BYTES)
MASSAGE INDICATOR BYTES (8 BYTES)
ENCRYPTED IP DATA

FIG. 7


START
PRECOMPUTE ALL SHARED SECRETS FOR EACH CLOSED GROUP NODE
DELETE j (J'S DH SECRET)
WAIT FOR PKTS OR TIMEOUT
TIME TO CHANGE N_j
$N_j = N_j + 1$
COMPUTE $N_j + 1^{th}$ SECRET ij FROM N_j^{th} SECRET
DELETE N_j^{th} SECRET
RECEIVE PKT IN CONTEXT N_i
COMPUTE $[K_{ij}]_{N_i}$
DECRYPT PACKET P
IS P A VALID ENCRYPTED PKT IN CONTEXT N_i ?
NO
DISCARD PKT
YES
DELETE N^{th} SECRET
$N_j = N_i$